# Image Recognition Using Artificial Neural Networks with Particle Swarm Optimization Based on Hardware FPGA

**Assist. Prof. Dr. Hanan A. R. Akkar*** **M.Sc. Muthana Khallil Ibrahim**
**Department of Electrical Engineering**
**University of Technology**

## Abstract:

**In this paper, a medical image recognition using Artificial Neural Networks (ANN) trained by Particle Swarm Optimization based on hardware implementation of Field Programmable Gate Array (FPGA) is presented, where the adaption of the Artificial Neural Network (ANN) weights using Particle Swarm Optimization (PSO) was proposed as a mechanism to improve the performance of ANN. Also in this paper, Hardware Design of ANN platform (HDANN) is proposed to evolve the architecture ANN circuits using FPGA-spartan3 board (XSA-3S1000).**

**The HDANN design platform creates ANN design files using WebPACKTM ISE10.1 program, which are converted into device-dependent programming files for eventual downloading into FPGA device by using GXSLOAD program from the XSTOOLS programs.**

**Keywords: ANN, PSO, FPGA, Medical Image.**

## 1- Introduction

**ANNs are computational networks which attempt to simulate the networks of the nerve cell (neurons) of the biological (human or animal) central nervous system. This simulation is a grass cell by cell simulation. It borrows from the neurophysiologic knowledge of the biological neurons and of networks of such biological neurons. It thus differs from conventional (digital or analog) computing machines that serve to replace, enhance or speed up human brain computations without regard to the organization of the computing elements and of their networking [3]. The primary significance for an ANN is the ability of the network to learn from its environment and to improve its performance through learning. The Back-propagation (Bp) algorithm is commonly used learning algorithm for Feed Forward Artificial Neural Networks. Bp algorithm is used in ANN learning process for supervised or associative learning [1].**

**The role of Artificial Neural Network (ANN) in the present world applications is gradually increasing and faster algorithms are being developed for training Neural Networks (NNs). For all these algorithms storage and computational requirements are different, some of these algorithms are good for pattern recognition and others for function approximation, but they have drawbacks in one way or another. Certain training algorithms are suitable for some types of applications only. It's difficult to find a particular training algorithm that is the best for all applications under all conditions and [2].**

**PSO is a population based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as genetic algorithm. The system is initialized with a population of random solutions and search for optima by updating generations. In PSO the potential solution is called particles; fly through the problem space by following the current optimum particles. During the past few years, PSO has been shown successful for many applications, especially in NNs training. PSO is applied to find optimal weights for the Feed Forward Artificial Neural Networks (FFANNs) [3].**

**FPGAs are digital integrated circuits that contain configurable (programmable) blocks of logic along with configurable interconnections between these blocks. Depending on the way in which FPGA are implemented, some FPGAs may only be programmed a single time, while others may be reprogrammed over and over again [4].**

## 2- Artificial Intelligent

**ANNs are computational networks which attempt to simulate, in a gross manner, the networks of nerve cell (neurons) of the biological central nervous system. ANNs are an attempt to create machines that work in a similar way to the human brain by building these machines using components that behave like biological neurons, called artificial neurons. The function of an ANN is to produce an output pattern when presented with an input pattern [5].**

## 2-1 Particle Swarm Optimizations

**PSO is an emerging population based optimization method, a parallel evolutionary computation technique originally designed by Kennedy and Eberhart in 1995. PSO is a kind of random search algorithm that simulates nature evolutionary process and performs good characteristic in solving many difficult optimization problems. The basic concept of PSO basically comes from a large number of birds flying randomly and looking for food together. Each bird is an individual called a particle. As the bird looking for food, the particles fly in a multidimensional search space looking for the optimal solution. All the particles can remember their own flying experience. According to the cognitive memory. The particles can adjust their position moving toward their global best position or their neighbor's local best position. PSO has a few parameters to adjust, so that it's convenient to make the parameters reach to the optimum values [6].**

**The swarm of particles initialized with a population of random candidate solutions move through the d-dimension problem space to search the new solutions. The fitness, f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector presenti (i is the index of the particle), and a velocity represented by a velocity-vector velocityi. After every iteration the best position-vector among the swarm so is stored in a vector.**

**The update of the velocity from the previous velocity to the new velocity is determined by equation (1). The new position is then determined by the sum of the previous position and the new velocity by equation (2) [7].**

$$\text{Velocityi}(t) = w^* \text{velocityi}(t-1) + c1\text{rand1}(\text{pbesti}(t)) - \text{positioni}(t-1) + c2\text{rand2}(\text{gbesti} - \text{positionti}(t-1)) \quad (1)$$

$$\text{positioni}(t) = \text{positioni}(t-1) + \text{velocityi}(t) \quad (2)$$

**Where w is inertia weight, c1 and c2 are acceleration constants and r is a random function in the range [0,1]. For equation (2.1), the first part represents the inertia of pervious velocity; the second part is the "cognition" part, which represents the private thinking by itself; the third part is the "social" part, which represents the cooperation among the particles. pbesti is the personal best position recorded by particle i, while gbesti is the global best position obtained by any particle in the population [8].**

## 3 - Field Programmable Gate Array (FPGA)

**FPGA are prefabricated silicon devices that can be electrically programmed to become almost any kind of digital circuit or system. They provide a number of computing advantages over fixed function ASIC technologies such as standard cells: ASICs typically take months to fabricate and cost hundreds of thousands to millions of dollars to obtain the first device; FPGAs are configured in less than a second, and often be reconfigured if a mistake is made, and cost anywhere from a few dollars to a few thousand dollars [9].**

**FPGA consists of an array of uncommitted Configurable Logic Block (CLB), programmable interconnects and Input Output Blocks (IOBs). FPGA architecture is dominated by programmable interconnects and CLBs which are relatively simple. This feature makes these devices for more flexible in terms of the range of designs that can be implemented with these devices[10].**

## 4- The Proposed Architecture of PSO Neuron

**PSO is one of the latest techniques that can be fitted into ANN. A swarm is made up of particles, where each particle has a position and a velocity. The idea of PSO in ANN is to get the best set of weight (or particle position) where several particles (problem solution) are trying to move or fly to get the best solution.**

**In PSONN, the position of each particle in swarm represents a set of weights for the current epoch or iteration. The dimensionality of each particle is the number of weights associated with the network. The particle moves within the weight space attempting to minimize learning error (or Mean Squared Error-MSE or Sum of Squared Error-SSE). Changing the position means updating the weight of the network in order to reduce the error of the current epoch. In each epoch, all the particles update their position by calculating the new velocity, which they use to move to the new position. The new position is a set of new weights used to obtain the new error. For PSO, the new weights are adapted even though no improvement is observed. This process is repeated for all the particles. The particle with the lowest error is considered as the global best particle so far. The training process continues until satisfactory error is achieved by the best particle or computational limits**

**are exceeded. When the training ends, the weights are used to calculate the classification error for the training patterns. The same set of weights is used then to test the network using the test patterns.**

## 5- Training ANN using BP& PSO for Medical Images

**Three-layer ANN is used to do the recognition on this paper for all datasets. The total number of neurons for every hidden layer is different depending on the classification problem. Number of input layer and output layer usually come from number of attribute and class attribute. However there is no appropriate standard rule or theory to determine the optimal number of hidden nodes.**

**In this work, trial and error has been used to determine number of hidden neurons. The activation function used to calculate output for each neuron is sigmoid activation (transfer function equation) except input neuron.**

**The color indices associated with image pixels are used as inputs, it is assumed that the ANN model could develop the ability to use other information, such as shapes, implicit in these data, these color pixel images are taken from the medical field, each image depicting only one Virus image , either a virus type 1 or virus type 2 ...etc. A total of three types of viruses images (human virus image , bird virus image, and varicella virus image) as shown in figure (2) are used for training purposes.**

**After the Red-Green-Blue (RGB) color images were obtained with size 160x160 pixels and then converted to indexed images based on a Gray Scale and requirement the size suitable for training the ANN. Figure(1) shows the original images for three types of viruses.**

**Figure (1) Original Images, for human influenza virus, bird influenza virus, and varicella virus.**

## 6- Simulation Results of Training ANN with PSO and BP

**Before training process, the parameters of training algorithm (PSO) are set to swarm size=30 particles, C1=C2=2, r1=0.8 and r2=0.2 and Φ=1.9, the initial weight (Φ) decreases from 1.8 to 1 linearly. The initial weights are randomly generated between [-15, 15] and biases between [-15,15] with maximum velocity (Vmax=10).**

**The problem dimension (total number of weights and biases) represents the size of search space in X-Y dimensions which is calculated from the relation below.**

$$\text{Dimension} = (\text{input}_{NN} * \text{hidden}) + (\text{hidden} * \text{output}_{NN}) + \text{hidden}_{bias} + \text{output}_{bias} \quad (3)$$

**The parameters of Back-propagation algorithm are set to the momentum coefficient α =0.9 and the learning rate η =0.54. The initial weights and biases are randomly generated between [-0.45, 0.45].**

**In the training process, three cases are discussed in this work as shown below. During training, the ANNs were presented with binary output data. There are only two output variable in the training data set. An output variable value of zeros are assigned to Human influenza virus image and a value of ones to varicella virus image and a value of zero and one to bird influenza virus image .**

**The ANN scheme trained by using PSO and BP is as shown in figure (2). There are 2 output variables in the training data, dependent on the size of image, in this case the input processing element is 100 neuron, and 100 hidden neurons based on trial and error.**

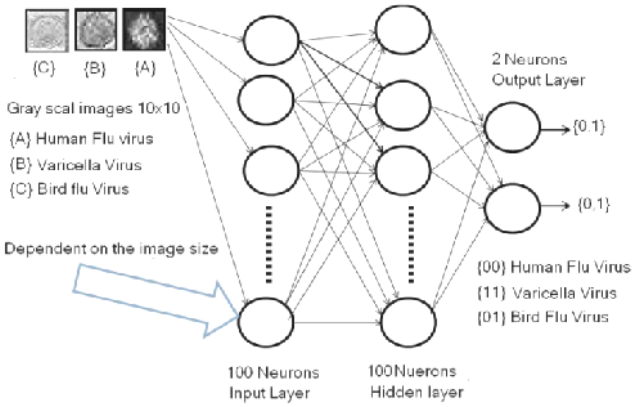
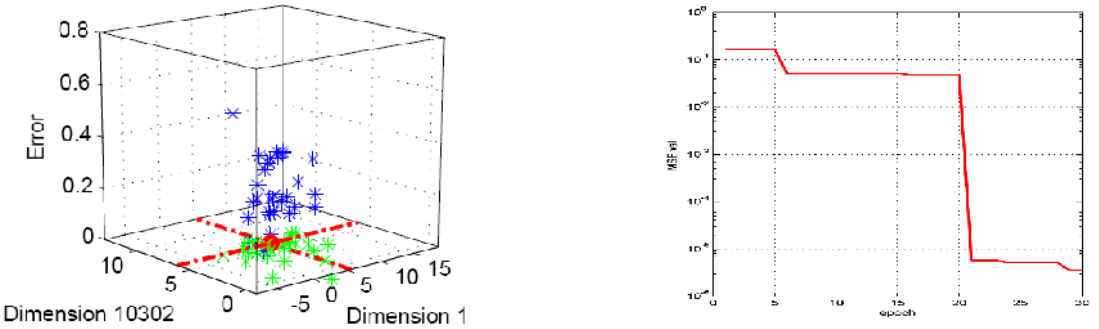



**Figure (2) Training ANN Using PSO and BP**

**In this case, the parameters of PSO algorithm are similar to the parameters mentioned in section 6. These numbers represent the size of the search space in X-Y dimensions. The problem dimensions (total number of weights and biases) are 10302, which are calculated from last relation that discussed in this section. The maximum number of iterations (epoch) = 100 and Mean Square Error (MSE)=10e-6.**

**The accuracy of performance and the performance for FNNPSO on three images dataset is shown in Figure (3) where the weights (particle positions) move towards best position.**




(b) (a)

**Figure (3) a-The performance of FNNPSO training. b- The accuracy of the performance.**

**The parameters of training algorithm (BP) are the default parameters. The maximum number of iterations (epoch)=100 and MSE=10e-6.**

**The performance of FNNBP on three images dataset with 100 hidden neurons is shown in Figure (4).**

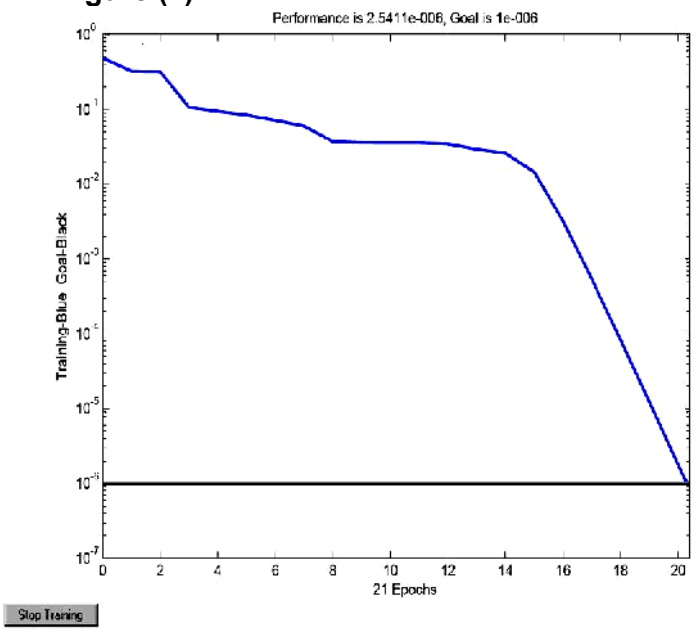



**Figure (4) the performance of FNNPSO training.**

**The simulation results of FNNPSO and FNNBP for three medical images training with 100 hidden neurons are shown in table (1).**

**Table (1) Results of FNNPSO and FNNBP of case one.**

| Parameters | FNNPSO | FNNBP |
|---|---|---|
| Learning Iterations | 30 | 20 |
| Error Convergence | 5.87e-006 | 2.5411e-6 |
| Convergence Time | 20.13sec | 1.8271e+003 sec. |
| No. of Initial Weights | 25 set | 1 set |
| Accuracy (%) | 99.98 | 99.5 |

From table (1), the results show that FNNPSO convergence time is very fast, where it takes 20.13 seconds at 30 iterations compared withFNNBP, where it takes 1.8271e+003 seconds at iteration 20 for overall learning process. Because the number of fitness evaluation (calculate minimum error for each bit and comparison with pbest). Both algorithms are converged using the minimum error criteria. For the correct accuracy percentage, it shows that FNNPSO result is better than FNNBP with 99.98 % compared to 99.5 %.

Processing in the recognition stage is similar to the classification stage, except that recognition stage also incorporates steps to match the input unknown images (noisy image) with those reference images in the database by neural network. The classification is achieved by train ANN. Samples used for testing include images with several types of noise (Gaussian noise and Salt and Pepper noise) between 1 to 10 dB and also ten images with different contrast and illumination conditions are generated for the samples.

**Table (2) Recognition performance on test images using PSOANN and BPANN**

| Training type | ANN structure | Recognition rate 3 attempt | Average of recognition |
|---|---|---|---|
| PSOANN | 100:100:2 | 95.35% | 91.4% |
| BPANN | 100:100:2 | 90.31% | 87.4% |

The table (2) shows that FFANN blocks are robust enough in handling the image variations. During recognition phase, Neural Network is explored for robust decision in the presence of wide variations. PSO exhibits the most favorable performance, on account of the fact that it has the lowest overall training time,and the highest recognition rates when compared to BP methods.

## 7- The Proposed Hardware Design of ANN Platform (HDANN)

The proposed Hardware Design of ANN platform (HDANN) is a circuit design platform built to evolve the architecture of ANN circuits using FPGA hardware. Therefore, the system must have software which can be downloaded into an FPGA. The basic hardware and software components of the proposed HDANN are shown in Figure (5). It consists of a computer and FPGA-spartan3 board (XSA-3S1000 Board).

The schematic-based-design is used to design the ANN circuits by using WebPACKTM ISE 10.1 program which are downloaded to an FPGA development board. The prototype board used in this work is XSA-3S1000 Board as shown in Figure (5). This board contains 1,000,000 gates Xilinx-Spartan3. FPGA in a 256 pin Ball Grid Array (BGA) package (XSA-3S1000), it is the main repository of programmable logic on the XSA-3S1000 Board and has a fixed frequency oscillator of 100 MHz

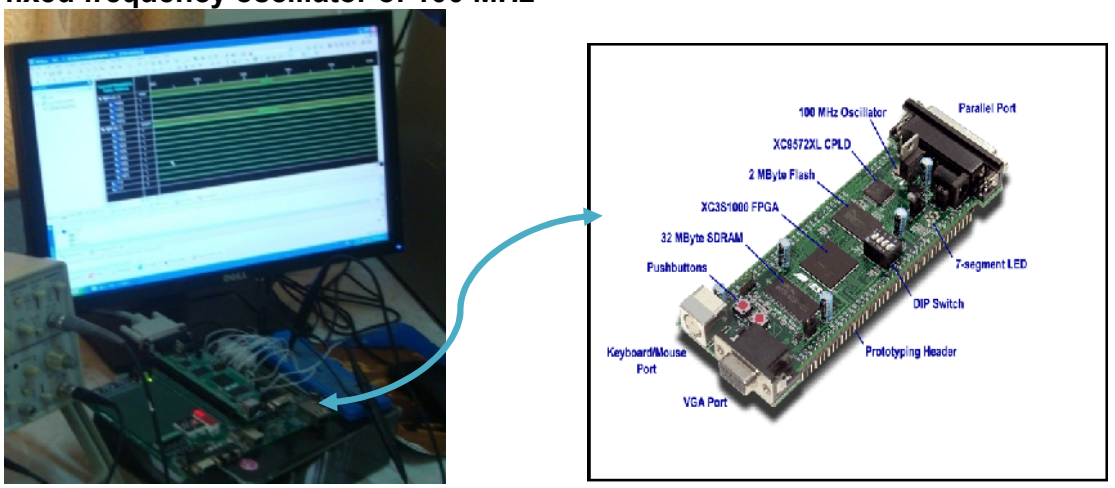



Figure (5) Basic hardware and software components of HDANN.

## 7-1 The Proposed Neuron Model

The neuron module is responsible for multiplying chosen inputs by static weight values, summing the results, and then producing the output.

Ten-input neuron module is shown in Figure (6), which consists of multiplier circuits, ADDSUB circuits, and comparator-high-low (compel) circuit as a hard-limiter function .The weights are blocked in the integer range of [-15, 15], four bits will represent the weight magnitude and a single bit represents the weight's sign . The bias range between [-15, 15], four bits will represent the bias magnitude and a last bit represents the bias's sign.

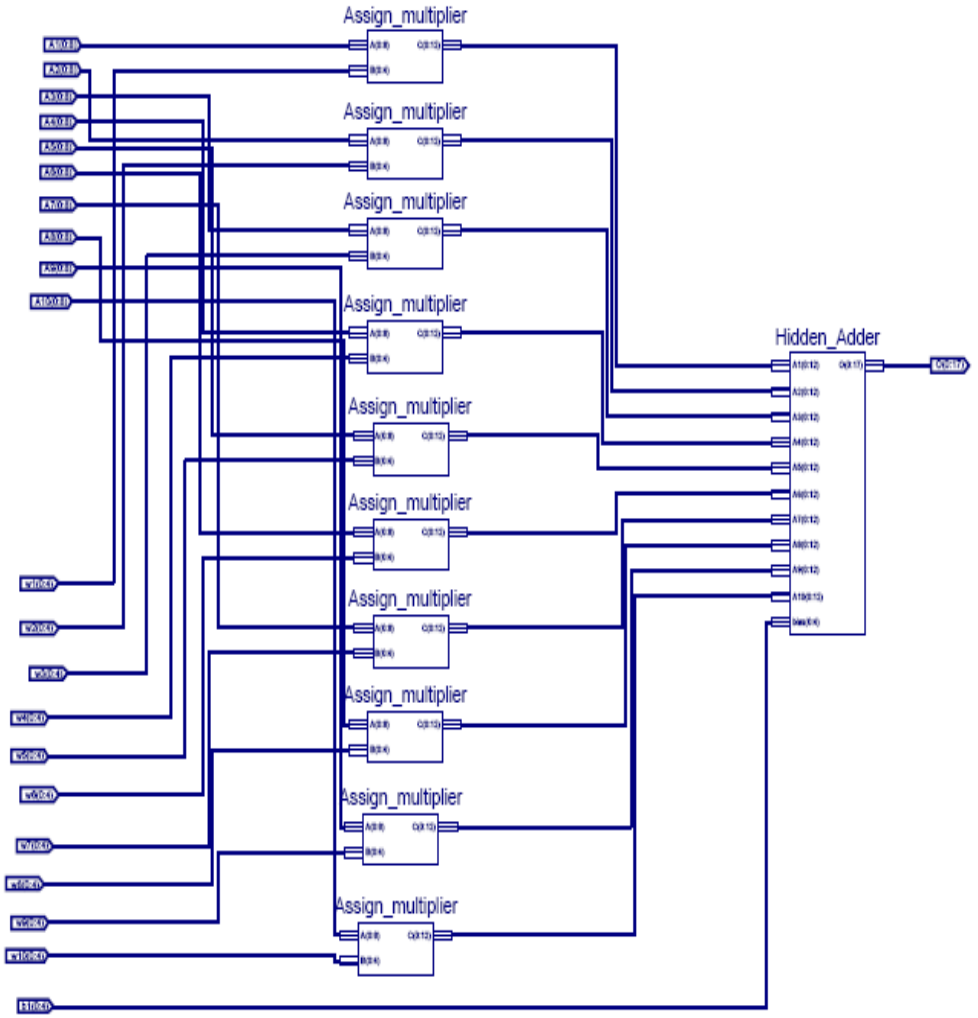



**Figure (6) the schematic of proposed ten inputs neuron module.**

**Based on the weight value for the ANN design, these weight values are multiplied with the inputs using multiplier circuit (MULTIPILER 8x5 Bits) as shown in Figure (7).**

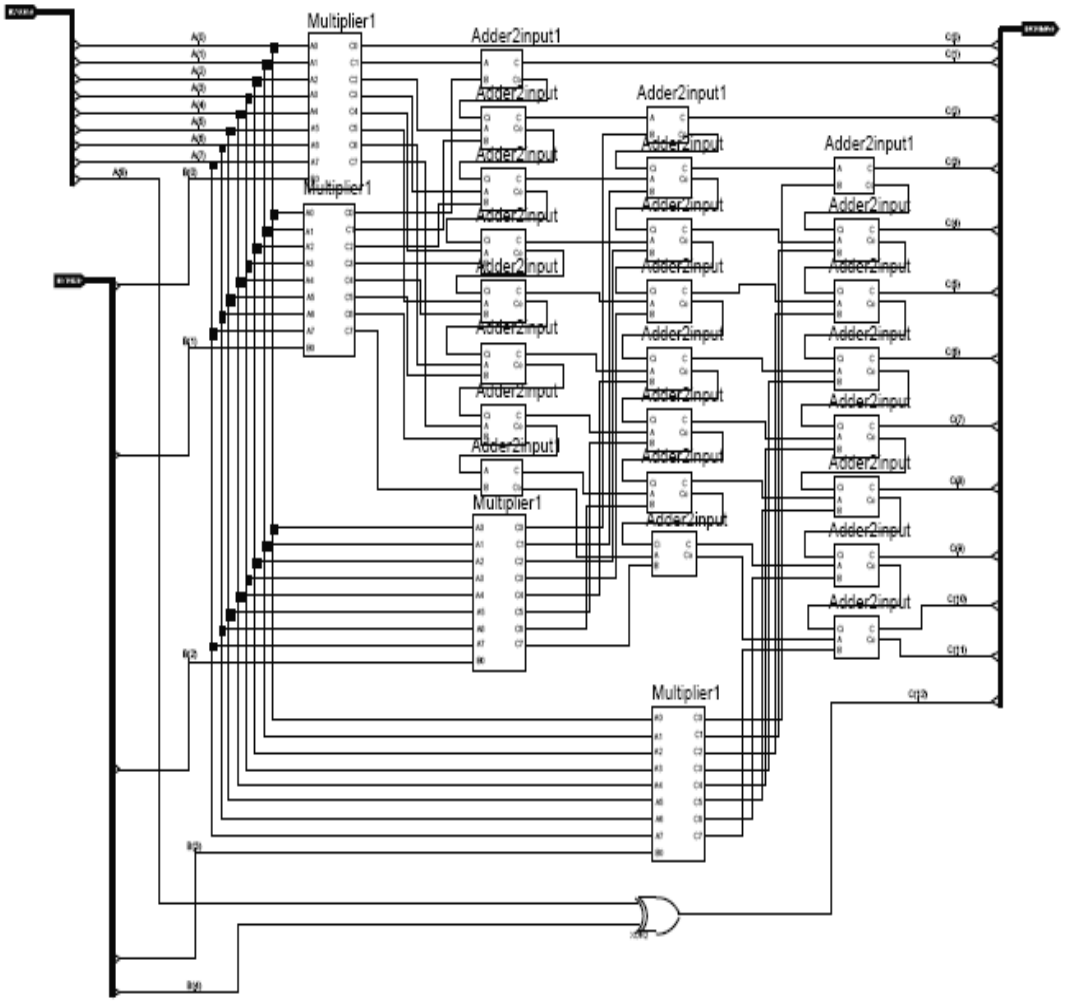



**Figure (7) Design input multiplier (8 bit Gray scale input x 5 bit weight).**

**This structure will be repeated for each input. The product produced by the multiplication process will be added or subtracted according to the weights signs using special designed Adder/ Subtracter with sign digital circuit. Figure (8) shows ADDSUB-12 bit circuit. The final result after ADDSUB circuit is threshold by using (comphl) circuit that is shown in Figure (9). This output can be fed to other neurons based on the connectivity in the ANN design. The number of neuron modules in the circuit depends on the number of neurons specified for the ANN design.**

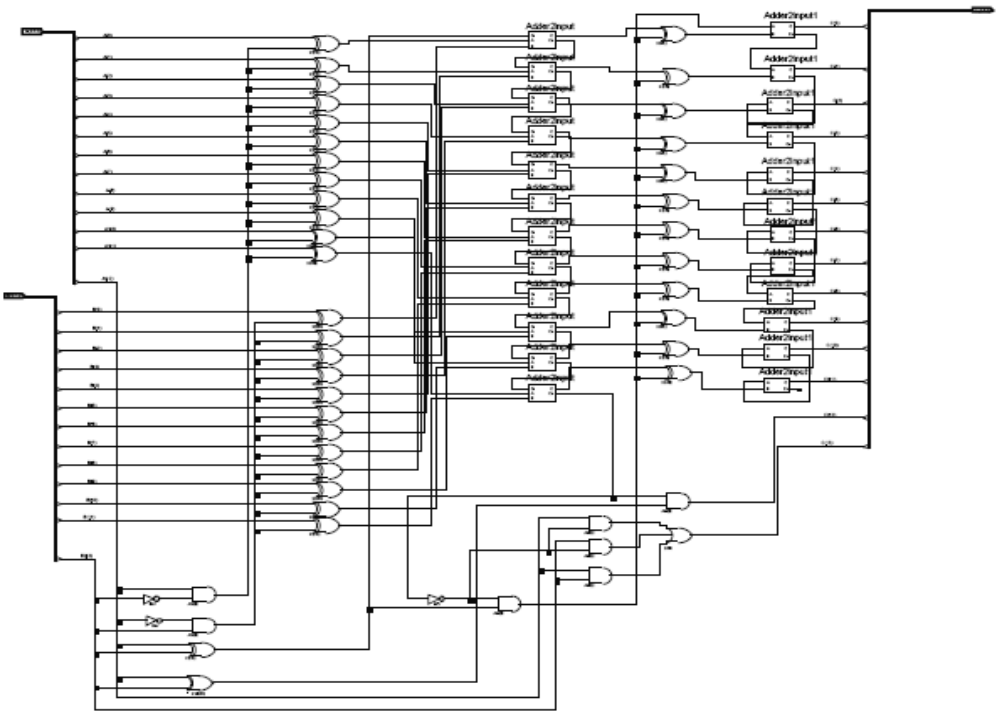


**Figure (8) Schematic of the proposed 12 Bit Adder with Bias 4 Bit.**

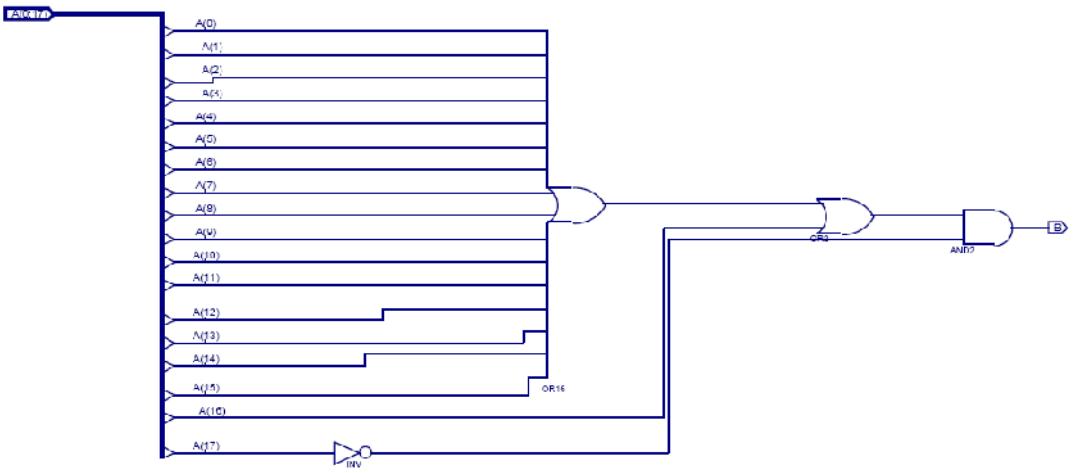
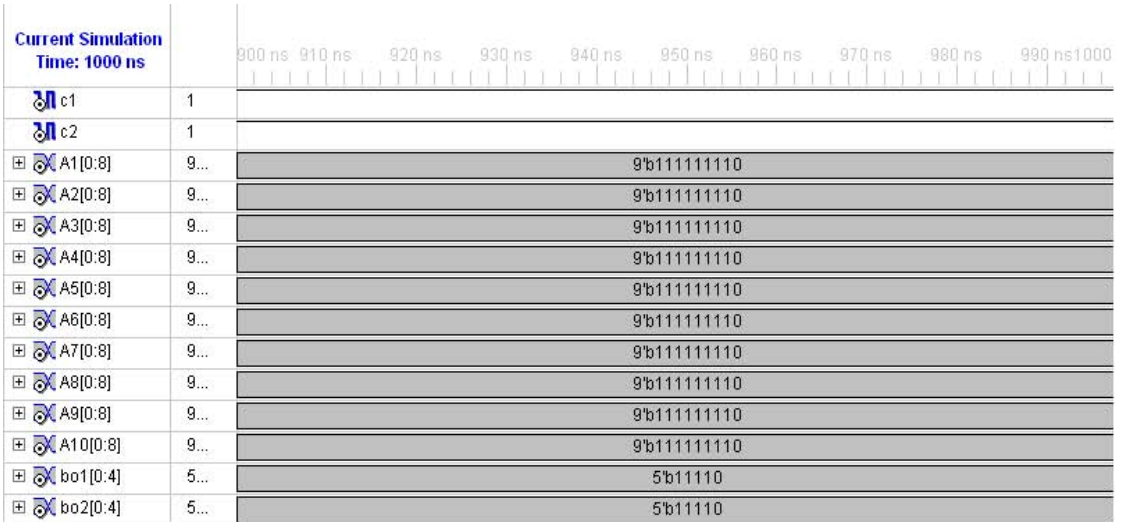
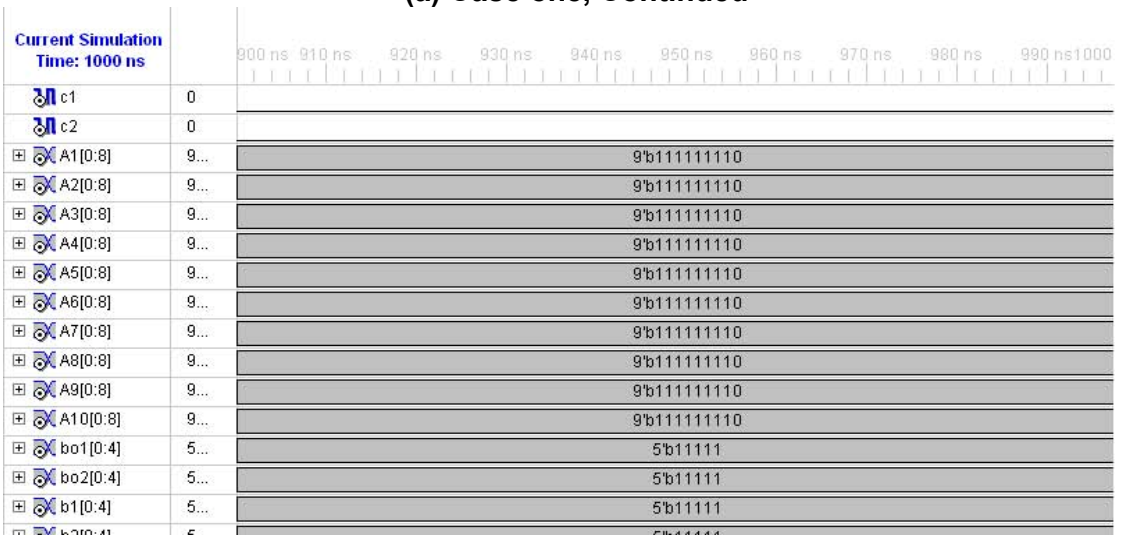



**Figure (9) Schematic of the proposed Comphl-18 bit module.**

**Design simulation: create a test bench waveform containing input stimulus which can be used to verify the functionality of the design module. The test bench waveform is a graphical view of a test bench. Figure (10) shows the simulation results of the ANN circuit of the proposed system.**




**(a) Case one, Continued**




**(b) Case two**

**Figure (10) The simulation results of the ANN circuit of the Proposed circuit.**

**Device Programming: downloading the bit-stream to the FPGA. Figure (11) shows the downloading of the bit-stream to the FPGA device (Spartan3- XSA-3S1000). After down loading the design into the board, a DC function generator has been applied to the input-pins of the ANN design while the output has been measured by an oscilloscope. Figure (12) shows the output data when various input values are applied to the network.**

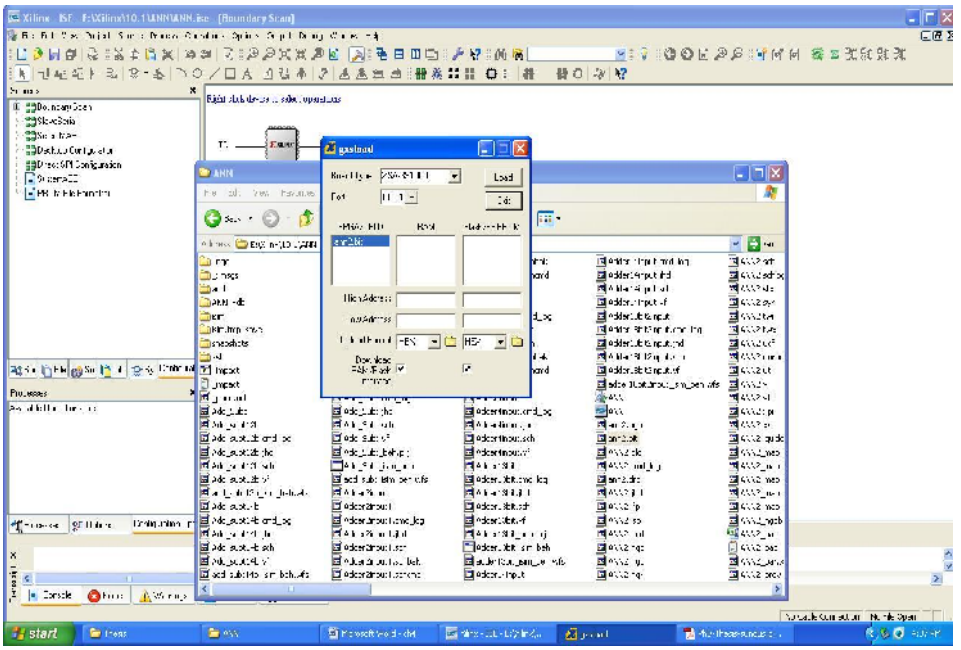
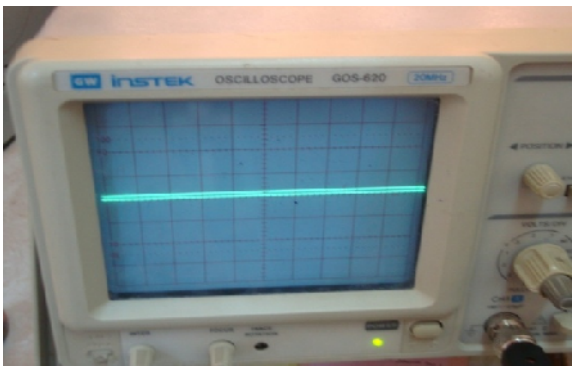
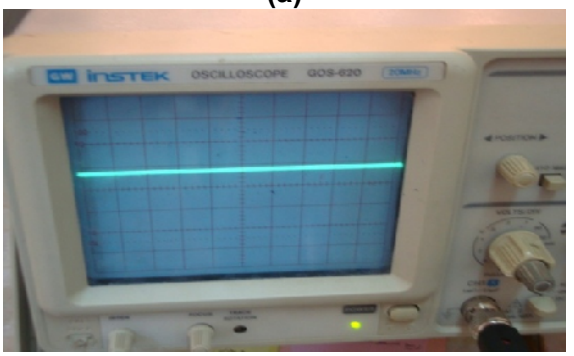


**Figure (11) The downloading of the bit-stream to the FPGA**




**(a)**




**(b)**

**Figure (12) The output data when the input to the ANN is applied for : (a) Case1 (b) Case2**

## 8 .Conclusions

**Referring to the objectives, the experiments are carried out to analyze the optimization algorithm called PSO that is applied on FNN to explore the classification accuracy and convergence time compared with Back-propagation Neural Network. Based on the results, it is clear that FNNPSO is better than FNNBP in terms of multi-starting points (initial weights), size of search space and accuracy of results.**

**In FNNPSO, network architecture and selection of network parameters for the dataset influence the convergence, and the performance of network learning. Reducing the hidden nodes minimizes the number of weights (position) and reduces the problem dimensions which could be taken to obtain the same results in less time, but not from the first second or even 10th attempts according to the networks input, output and complexity. Conversely, the results in table (1) indicate that PSO algorithm is more robust with more consistent convergence characteristics. Thus, PSO algorithm can be applied to wide-range nonlinear optimization problems, such as medical images recognition with reliable performance. In addition, PSO and similar optimization algorithms do not add a considerable computational burden.**

**In this work a hardware implementation of neural networks by FPGAs has been proposed. The proposed network architecture modular, being possible to easily increase or decrease the number of neurons as well as layers. FPGAs can be used for portable, modular, and reconfigurable hardware solutions for neural networks, which have been mostly used to be realized on computers until now. FPGA technologies are fairly new and rapidly advancing in gate count and speed. FPGA is the best candidates in neural network implementations among the other alternatives.**

**The neural network that was implemented on the FPGA is restricted in complexity depending upon the availability of space on the FPGA. Also, the system that is implemented should have a relatively small number of low digit inputs with a few digital outputs. There are drawbacks with using Xilinx FPGA (Spartan3 XSA-3S1000) such as, it has finite I\O ports, and therefore the neural network is represented with minimum inputs as possible to make the hardware implementation on Spartan3 XSA-3S1000 possible.**

## 9 References


1- T. L. Fine **"Feed forward Neural Network Methodology",** Springer publisher, ISBN: 0-387-98745-2, 1999.

2- T. Su, J. Jhang and G. Hou, **"A Hybrid Artificial Neural network and Particle Swarm Optimization for Function Approximation"** International Journal of Innovative Computing, Information and Control, Vol. 4, No. 9, September, 2008.

3- A. Lazinica**, "Particle Swarm Optimization",** In-Tech publisher, ISBN: 978-953-7619-48-0, January, 2009.

4- C. Maxfield**, "The Design Warriors Guide to FPGAs:** devices, tools, and flows", Newness publisher, ISBN: 0-7506-7604-3, 2004.

5- D.D. Earl **"Development of an FPGA-Based Hardware Evaluation System for Use with GA-Designed Artificial Neural Networks",** PhD thesis, The University of Tennessee, Knoxville, May 2004.

6- L. Wang, X. Wang, J. Fu and L. Zhen**, "A Novel Probability Binary Particle Swarm Optimization Algorithm and It's Application",** Academy Publisher, Journal of Software, Vol. 3, No. 9, PP. 28-35, December, 2008.

7- S. Sumathi, surekha p**. "Computational Intelligence paradigms ",** theory and applications using MATLAB, Taylor and Francis Group, LLC ISBN 978-1-4398-0902-0, 2010

8- K. E. Parsopulos and M. N. Vrahatis**, "Recent Approaches to Global Optimization Problems through Particle Swarm Optimization",** Kluwer Academic, Natural Computing Conference, PP. 235-306, 2002.

9- R. Woods, J. Mcallister, G. Light body and Y. Yi, **"FPGA-Based Implementation of Signal Processing System",** Wiley, ISBN: 978-0-470-03009-7, 2008.

**10-I.** Kuon, R. Tessier and J. Rose, **"FPGA Architecture: Survey and Challenges",** Wiley, ISBN: 978-1-60198-126-4, 2008.

# تمييز الصور بواسطة الشبكات العصبيةِ الاصطناعية وامثلية الحشد الجزيئي

# المنفذة عمليا بمصفوفة البوابات القابلة للبرمجة

أ.م.د.حنان عبد الرضا عكار
الجامعة التكنولوجية/ قسم الهندسة الكهربائية

م. م. مثنى خليل ابراهيم
الجامعة التكنولوجية/ قسم الهندسة الكهربائية

## المستخلص

تم في هذا البحث تدريب الشبكات العصبية الاصطناعية باستخدام أمثلية الحشد الجزيئي لتمييز الصور الطبية وتنفيذها عمليا بواسطة كارت مصفوفة البوابات القابلة للبرمجة FPGA وذلك لتحسين أداء الشبكات العصبية الاصطناعية. أيضا تم في هذا البحث استخدام الكارت العملي لمصفوفة البوابات القابلة للبرمجة FPGA لتمثيل ANN المدربة باستخدام PSO، وذلك بسبب السرعة و قابلية إعادة البرمجة. يمكن للـ FPGA أن يدعم إعادة التشكيل او التصميم اللازمة لتمثيل الشبكة العصبية. تم التنفيذ العملي للشبكات العصبية (HDANN) باستخدام FPGA-spartan3 board (XSA3S1000).

تم في هذا العمل اقتراح (HDANN) لتمثيل ANN بإستخدام FPGA-spartan3 board (XSA-3S1000). بإستخدام HDANN حيث يتم إنشاء الملفات الخاصة بتصميم ANN بإستخدام برنامج ISE 10.1 WebPACK$^{TM}$، التي يتم تحويلها إلى ملفات البرمجة التي يعتمد عليها في نهاية المطاف لتحميلها الى FPGA بإستخدام برنامج GXSLOAD من مجموعة برامج XSTOOLS.